Generative Adversarial Source Separation

Cem Subakan, Paris Smaragdis

University of Illinois at Urbana-Champaign

April 17’th, 2018
Generative Modeling
Generative Modeling
Source Separation

Source 1

Source 2

Mixture
Source Separation

Source 1

Source 2

Mixture

Estimate for Source 1

Estimate for Source 2
Motivations for using GANs in source separation

- Generative Adversarial Networks (GANs) are a way to learn generative models.
  - GANs learn to generate data items that look like the training data.

- GANs can therefore potentially learn a distribution over each audio source.
Motivations for using GANs in source separation

- Generative Adversarial Networks (GANs) are a way to learn generative models.
  - GANs learn to generate data items that look like the training data.
- GANs can therefore potentially learn a distribution over each audio source.
- More technically, GANs learn “implicit generative” models which do not specify an output noise distribution.
Non-Negative Matrix Factorization

\( X \approx WH \)

- \( X \in \mathbb{R}^{L \times T} \rightarrow \text{Input Spectrogram} \)
- \( W \in \mathbb{R}^{L \times K} \rightarrow \text{Frequency Templates} \)
- \( H \in \mathbb{R}^{K \times T} \rightarrow \text{Activations} \)

**Learning:**

\[
\min_{W,H} d(X\|WH)
\]

- \( d(\cdot\|\cdot) \rightarrow \text{Divergence Measure}. \) E.g. KL-divergence, Euclidean distance. Choice of which effects results.
Non-Negative Matrix Factorization

\( X \approx WH \)

- \( X \in \mathbb{R}^{L \times T} \rightarrow \text{Input Spectrogram} \)
- \( W \in \mathbb{R}^{L \times K} \rightarrow \text{Frequency Templates} \)
- \( H \in \mathbb{R}^{K \times T} \rightarrow \text{Activations} \)

Learning:

\[
\min_{W,H} d(X \| WH)
\]

- \( d(\cdot \| \cdot) \rightarrow \text{Divergence Measure}. \) E.g. KL-divergence, Euclidean distance. Choice of which effects results.

**KL NMF**

![KL NMF Diagram]

**Euclidean NMF**

![Euclidean NMF Diagram]

We used the same parameter initialization for both costs.
Generative Model Generalization for NMF

$X_t \sim p_{out}(X_t; WH_t)$,

$$\max_{W,H} \log \prod_t p_{out}(X_t; WH_t) \propto \min_{W,H} \sum_t d(X_t \| WH_t)$$

$p_{out}(\cdot; \cdot) \rightarrow$ output distribution that corresponds to the divergence measure $d(\cdot \| \cdot)$. E.g. Poisson for un-normalized KL divergence, Gaussian for Euclidean distance.
Generative Model Generalization for NMF

- $X_t \sim p_{out}(X_t; WH_t)$,

$$\max_{W,H} \log \prod_{t} p_{out}(X_t; WH_t)$$

$$\propto \min_{W,H} \sum_{t} d(X_t \| WH_t)$$

- $p_{out}(\cdot; \cdot)$ → output distribution that corresponds to the divergence measure $d(\cdot \| \cdot)$. E.g. Poisson for un-normalized KL divergence, Gaussian for Euclidean distance.

- Our goal in this paper is to use a generative model which does not specify $p_{out}(\cdot; \cdot)$. (or equivalently $d(\cdot \| \cdot)$).
NMF Generalizations

Standard NMF

\[ X_t \sim p_{out}(x; WH_t) \]
NMF Generalizations

**Standard NMF**

\[ X_t \sim p_{\text{out}}(x; WH_t) \]

**Probabilistic NMF**

\[ H_t \sim p_{\text{prior}}(H_t) \]

\[ X_t | H_t \sim p_{\text{out}}(x; WH_t) \]
## NMF Generalizations

<table>
<thead>
<tr>
<th></th>
<th>Probabilistic NMF</th>
<th>Probabilistic Neural-Net NMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard NMF</td>
<td>$X_t \sim p_{\text{out}}(x; WH_t)$</td>
<td>$H_t \sim p_{\text{prior}}(H_t)$</td>
</tr>
<tr>
<td></td>
<td>$X_t</td>
<td>H_t \sim p_{\text{out}}(x; WH_t)$</td>
</tr>
</tbody>
</table>
## NMF Generalizations

<table>
<thead>
<tr>
<th>Standard NMF</th>
<th>Probabilistic NMF</th>
<th>Probabilistic Neural-Net NMF</th>
<th>Implicit Density Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_t \sim p_{\text{out}}(x; WH_t)$</td>
<td>$H_t \sim p_{\text{prior}}(H_t)$</td>
<td>$H_t \sim p_{\text{prior}}(H_t)$</td>
<td>$H_t \sim p_{\text{base}}(H_t)$</td>
</tr>
<tr>
<td>$X_t</td>
<td>H_t \sim p_{\text{out}}(x; WH_t)$</td>
<td>$X_t</td>
<td>H_t \sim p_{\text{out}}(X_t; f_\theta(H_t))$</td>
</tr>
</tbody>
</table>

- Where implicit generative models define a model distribution via a deterministic transformation $f_\theta(H_t)$ of a base distribution $p_{\text{base}}(H_t)$.
- Instead of hand picking $p_{\text{out}}(.)$, we can use an implicit generative model, and train it via adversarial training.
# NMF Generalizations

<table>
<thead>
<tr>
<th>Standard NMF</th>
<th>Probabilistic NMF</th>
<th>Probabilistic Neural-Net NMF</th>
<th>Implicit Density Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_t \sim p_{\text{out}}(x; WH_t)$</td>
<td>$H_t \sim p_{\text{prior}}(H_t)$</td>
<td>$H_t \sim p_{\text{prior}}(H_t)$</td>
<td>$H_t \sim p_{\text{base}}(H_t)$</td>
</tr>
<tr>
<td>$X_t</td>
<td>H_t \sim p_{\text{out}}(x; WH_t)$</td>
<td>$X_t</td>
<td>H_t \sim p_{\text{out}}(X_t; f_\theta(H_t))$</td>
</tr>
</tbody>
</table>

- Where implicit generative models define a model distribution via a deterministic transformation $f_\theta(H_t)$ of a base distribution $p_{\text{base}}(H_t)$.
- Instead of hand picking $p_{\text{out}}(.)$, we can use an implicit generative model, and train it via adversarial training.

**ML objective**

$$\max_{\theta, H} \sum_t \log p_{\text{out}}(X_t; f_\theta(H_t)),$$

**Adversarial training objective**

$$\max_\xi \min_\theta \sum_t \log D_\xi(X_t) + \sum_{t'} \log(1 - D_\xi(X_{t'})),$$

where $X_{t'} = f_\theta(H_{t'})$. 
<table>
<thead>
<tr>
<th>NMF Generalizations</th>
<th>Probabilistic NMF</th>
<th>Probabilistic Neural-Net NMF</th>
<th>Implicit Density Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard NMF</td>
<td>$X_t \sim p_{\text{out}}(x; WH_t)$</td>
<td>$H_t \sim p_{\text{prior}}(H_t)$</td>
<td>$H_t \sim p_{\text{base}}(H_t)$</td>
</tr>
<tr>
<td></td>
<td>$X_t</td>
<td>H_t \sim p_{\text{out}}(x; WH_t)$</td>
<td>$X_t</td>
</tr>
</tbody>
</table>

- Where implicit generative models define a model distribution via a deterministic transformation $f_\theta(H_t)$ of a base distribution $p_{\text{base}}(H_t)$.
- Instead of hand picking $p_{\text{out}}(.)$, we can use an implicit generative model, and train it via adversarial training.

**ML objective**

$$\max_{\theta, H} \sum_t \log p_{\text{out}}(X_t; f_\theta(H_t)),$$

**Adversarial training objective**

$$\max_\xi \min_{\theta} \sum_t \log D_\xi(X_t) + \sum_{t'} \log(1 - D_\xi(X_{t'})),$$

where $X_{t'} = f_\theta(H_{t'})$.

- The training goal in GANs is to generate samples $X_{t'}$ so that they are indistinguishable from training data $X_t$. 

---

7 / 14
Training GANs

- It is standard to use the following bi-level optimization procedure:

\[
\begin{align*}
\max_{{\xi}} & \sum_t \log D_{\xi}(X_t) + \sum_{t'} \log(1 - D_{\xi}(f_\theta(H_{t'}))) \\
\max_{{\theta}} & \sum_t \log D(f_\theta(H_t))
\end{align*}
\]
It is standard to use the following bi-level optimization procedure:

$$\max_{\xi} \sum_t \log D_\xi(X_t) + \sum_{t'} \log(1 - D_\xi(f_\theta(H_{t'})))$$

$$\max_{\theta} \sum_t \log D(f_\theta(H_t))$$

This is the original formulation, and tends to collapse on subset of the data distribution.

Wasserstein formulation:

$$\max_{\xi \in \mathcal{W}} \sum_t D_\xi(X_t) - \sum_{t'} D_\xi(f_\theta(H_{t'})))$$

$$\max_{\theta} \sum_t D(f_\theta(H_t))$$

Tends to have better gradient flow.
Generating Spectrogram Frames with a GAN

- Using GANs enables us to generate plausible spectrogram frames with implicit models.
We evaluate the validity of adversarial training with supervised generative source separation task.
We evaluate the validity of adversarial training with supervised generative source separation task.

First train the generative models for each source.

Learn \( p_{\text{model}}(X_1|\theta_1) \)

i.e. train \( f_{\theta_1}(.) \),

Learn \( p_{\text{model}}(X_2|\theta_2) \)

i.e. train \( f_{\theta_2}(.) \)

The corresponding generative model for the mixture:

\[
X_1 \sim p_{\text{model}}(X_1|\theta_1) \\
X_2 \sim p_{\text{model}}(X_2|\theta_2) \\
X_{\text{mix}|X_1, X_2} \sim p_{\text{out}}(X_{\text{mix}; X_1 + X_2})
\]
In test time, the source estimates are obtained via optimizing w.r.t. the network inputs.

\[
\hat{H}_1, \hat{H}_2 = \arg \max_{H_1, H_2} p_{\text{out}}(x_{\text{mix}}; f_{\theta_1}(H_1) + f_{\theta_2}(H_2))
\]

(This is the same test procedure when doing source separation with supervised NMF)
Generative Adversarial Source Separation

Train $f_{\theta_1}(\cdot), D_{\xi_1}$

Train $f_{\theta_2}(\cdot), D_{\xi_2}$

$\hat{H}^1, \hat{H}^2 = \arg\max_{H^1, H^2} p_{\text{out}}(X_{\text{mix}}; f_{\theta_1}(H^1) + f_{\theta_2}(H^2)) + \lambda \left( \sum_{k=1}^{2} D_{\xi_k}(f_{\theta_k}(H^k)) \right)$

Estimate for Source 1

Estimate for Source 2
Results

- **Dataset:** Male-female speaker mixtures from TIMIT dataset.
  - Training set: 9 utterances for each speaker.
  - Test set: Single sentence mixture at 0dB.
  - Evaluated for 25 pairs of speakers.

- **Evaluation:** BSS eval metrics. (SIR, SAR, SDR)

- We compare Wasserstein GAN, NMF, Variational Autoencoders, Denoising Autoencoder, GAN, all with a multilayer perceptron architecture.
Conclusions

▶ Using implicit generative models improves the model accuracy on a speech source separation task.
  ▶ Implicit generative models do not require specifying an output distribution.
  ▶ We learn to generate plausible spectrogram frames.
▶ Generative models which operate over sequences is a natural next step.
▶ Download our code from https://github.com/ycemsubakan/sourceseparation_misc, try it out yourself.